электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.772

Протокол HTTP, его расширения и надстройки

Золотов А.А., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Самохвалов Э.Н., к.т.н., профессор,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
gapyu@bmstu.ru

Сейчас довольно трудно представить нашу жизнь без интернета и интернет технологий. Хотелось бы рассказать про протокол HTTP – основополагающий протокол в современном интернете.

HTTP — распространённый протокол передачи данных, предназначался изначально для передачи гипертекстовых документов.

Также HTTP является протоколом прикладного уровня в стандартной модели OSI.

В протоколе HTTP основой является технология «клиент-сервер», которая предполагает существование потребителей (клиентов), инициирующих соединение и посылающих запрос, и поставщиков (серверов), ожидающих соединения для получения запроса, производящих определенные действия и возвращающих обратно сообщение с результатом.

В настоящее время доля HTTP трафика (для Америки) составила 46% от общего объема трафика во всем Интернете. Протокол HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня таких как: SOAP, XML-RPC, WebDAV. Основным объектом для данного протокола является ресурс, на который указывает URI в запросе клиента.

Обмен сообщениями идет по системе «запрос-ответ». Как известно, протокол HTTP не сохраняет своего состояния. Компоненты в протоколе HTTP могут самостоятельно сохранять информацию о состоянии, связанном с последними запросами и ответами. Браузер посылает запросы и может отслеживать задержки ответов. Сервер хранит IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не получает информации о предыдущих запросах и ответах, в нём нет внутренней поддержки состояния.

Теперь поговорим немного про программное обеспечение:

Все программное обеспечение для работы с данным протоколом делится на 3 большие категории:

- Серверы: как основные поставщики услуг хранения и обработки информации.
- Клиенты: конечные потребители услуг сервера.
- Прокси для выполнения транспортных служб.

Существует 3 версии протокола HTTP:

HTTP/0.9: В марте 1991 года Тим Бернес-Ли предложил данную версию протокола, как механизм для доступа к различным документам в Интернете. Но впервые он был использован в январе 1992. Спецификация протокола упорядочила и стандартизировала правила взаимодействия между клиентами и серверами HTTP, а также было введено чуткое разделение функций между этими компонентами.

HTTP/1.0: в мае 1996 года выпущен документ RFC-1945 для практической реализации данного протокола. Именно данный документ стал основной базой для изменения протокола HTTP.

HTTP/1.1: в июне 1999 года была предложена очередная версия протокола. В ней присутствовал ряд нововведений и новшеств, а именно: режим постоянного соединения. TCP-соединение остается открытым после отправки ответа на запрос, что позволяет персональным ЭВМ посылать несколько запросов за одно соединение. Клиент (персональная ЭВМ) посылает информацию о наименовании хоста, к которому он обращается. Именно это сделало возможной простую организацию виртуального хостинга.

Протокол HTTP/1.1 является текущей версией протокола.

Каждое HTTP- сообщение состоит из 3 частей:

- Стартовая строка – определяет тип сообщения.
- Заголовки – характеризует тело сообщения.
- Тело сообщения – непосредственно данные сообщения.

Тело и заголовки могут отсутствовать, но стартовая строка присутствует всегда, т.к. указывает на тип запроса/ответа.

Стартовые строки различаются ответа и запроса. Строка запроса выглядит так:

GET URI — для версии протокола 0.9.

Метод URI HTTP/Версия — для остальных версий.

Например:

GET /wiki/HTTP HTTP/1.0

Host: ru.wikipedia.org

Ответ может выглядеть так:

HTTP/1.0 200 OK

200 это код состояние и он говорит о том, страница нашлась и ответ передан клиенту.

Метод HTTP – последовательность из любых символов, кроме управляющих и разделительных, указывающая на основную операцию над ресурсом. Чаще всего, метод представляет собой короткое слово на английском языке.

Каждый сервер обязан поддерживать как минимум два метода GET и HEAD. Кроме этих методов сервер, как правило, поддерживает метод POST.

Метод GET используется для запроса содержимого ресурса. Также с помощью метода GET можно запустить какой-нибудь процесс. В этом случае в теле ответного сообщения будет содержаться информация о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса после знака «?».

GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Кроме стандартного метода GET различают еще частичный и условный GET.

Метод HEAD похож на метод GET, за исключением того, что тело сообщения отсутствует. Данные запросы предназначены для извлечения метаданных, валидации URL и других важных операций.

Метод POST используется для передачи данных пользователя заданному ресурсу. Например, в блогах, социальных сетях посетители могут вводить свои комментарии к записям и фотографиям в HTML-форму, после эти данные передаются на сервер и он особыми методами и функциями помещает введенный текст на HTML-страницу. При этом данные, передаваемые на сервер, включаются в тело запроса. Аналогичным образом с помощью данного метода можно загружать файлы на сервер

Метод POST, в отличие от метода GET не считается идемпотентным, т.е. на одинаковые запросы, могут быть возвращены совершенно разные ответы.

Хотелось бы подробней остановиться на кодах состояния для протокола HTTP.

Всю структуру кодов состояния покажет данная концептуальная карта на Рис.1.

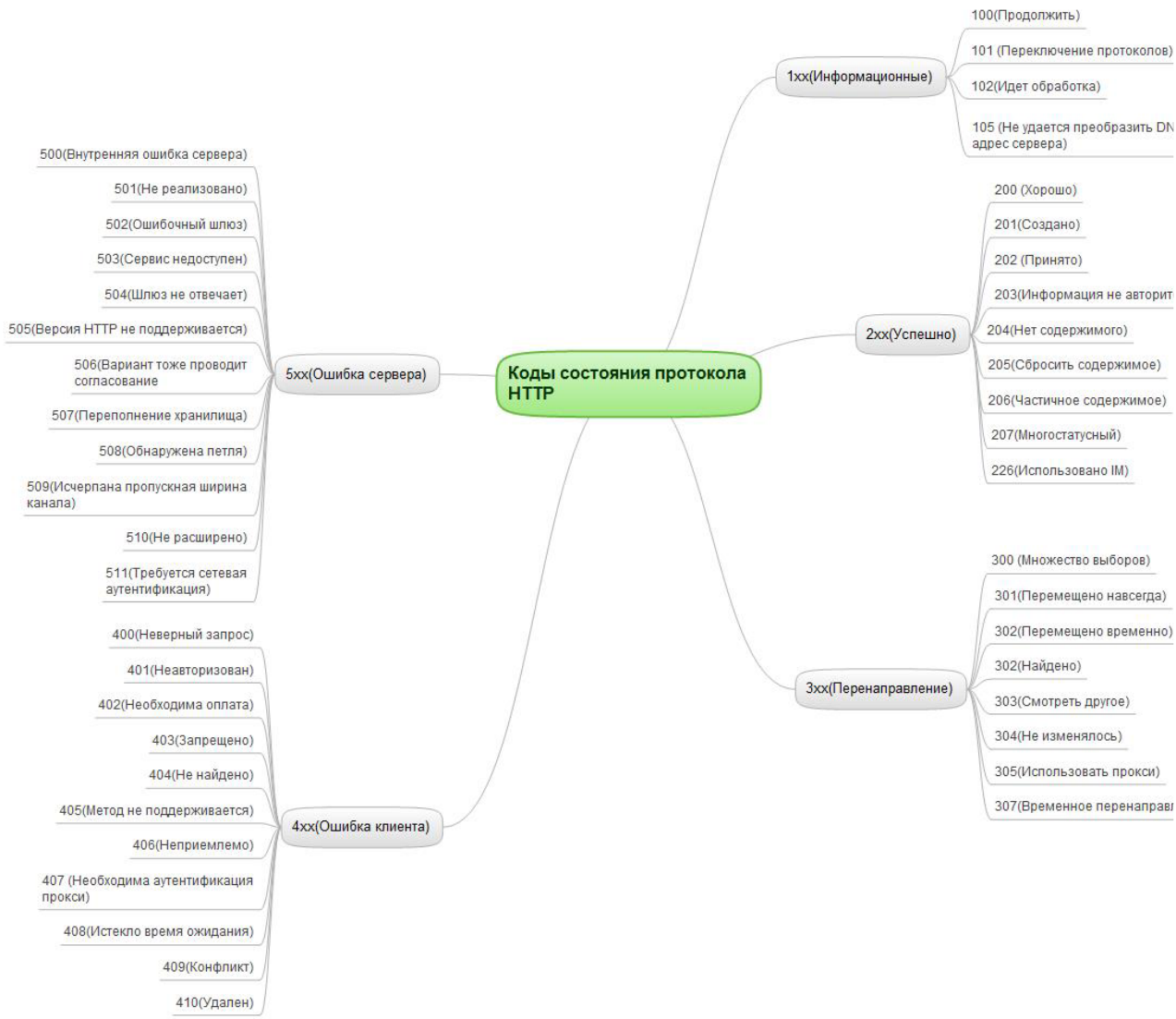

Рис. 1. Коды состояний протокола HTTP

Код состояния – составная часть первой строки ответа при запросах по данному протоколу. Он представляет собой число из трех цифр. Первая цифра указывает на класс состояния. 2 остальные цифры указывает на конкретное состояние протокола. За кодом, как правило, следует отделенная пробелом поясняющая английская фраза.

Клиент узнает по коду о результате и определяет, какие ему действия предпринять дальше. Набор кодов состояния является стандартным, и они описаны в соответствующих документах RFC. Введение дополнительных и новых допустимо только после согласования их с IETF (Инженерный совет Интернета).

Как видно из концептуальной карты все коды состояний разделены на 5 больших групп:

- Информационные: в этот класс отнесены коды, информирующие пользователя персональной ЭВМ о процессе передачи.

- Успех: сообщения этого класса несут информацию о случаях успешной обработки клиентского запроса. В зависимости от статуса запроса сервер может передать заголовки и некоторое тело сообщения.
- Коды класса 3xx информируют клиента, что для дальнейшего выполнения операции необходимо сделать другой запрос. Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся к перенаправлениям. Также в заголовке Location указывается адрес, по которому следует сделать перенаправление. При в целевом URI допускается использование фрагментов.
- Класс кодов 4xx используется для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть сообщение, в теле которого содержится гипертекстовое пояснение для пользователя.
- Коды 5xx используются в случаях неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме метода HEAD, сервер должен включать в объяснение в тело сообщения, которое клиент отобразит пользователю.

Все выше сказанное относится к структуре протокола, теперь же речь пойдет об основных механизмах протокола и принципах его работы.

Частичные GET: HTTP протокол позволяет запросить не все содержимое ресурса сразу, а только определенный (указанный) фрагмент. Данный механизм в основном реализован для загрузки больших файлов (по частям).

Также, некоторые программы скачивают заголовок архива, выводят внутреннюю структуру пользователю, а потом уже запрашивают фрагменты с указанными элементами архива.

Для получения какого-либо фрагмента клиент посылает на сервер запрос с заголовком Range (диапазон) и указывает в нем диапазоны в байтах (например:152000-256000).

Если сервер не понимает или не принимает механизм частичных GET, то он вернет код 200, как при обычном GET. В случае успешного запроса он вернет код 206 (частичный контент).

Сами фрагменты могут быть переданы двумя способами:

- Заголовок Content-Range с указанием байтовых диапазонов помещается в ответ и передается клиенту (персональной ЭВМ). Далее, указанные фрагменты помещаются в тело сообщения друг за другом.

- Сервер в ответе указывает тип multipart/byteranges для содержимого и передает фрагменты (частями). Также сервер обязан указать соответствующий Content-Range для каждого элемента.

Условные GET: метод GET меняется на «условный GET», если в сообщении содержится поле «If-Modified-Since». В ответ на такой запрос GET, тело запрашиваемого ресурса передается только, если в нем произошли какие-то изменения/добавления после указанной даты.

Алгоритм определения этого включает в себя следующие возможные случаи:

- В случае отличия кода, помещенного в тело ответа, от «200 OK», или указанная дата, в поле заголовка «If-Modified-Since» некорректна, то ответ аналогичен ответу на обычный запрос GET.
- Если после полученной в запросе даты ресурс изменялся или дополнялся, то ответ будет также идентичен ответу на обычный запрос GET.
- Если ресурс не изменялся после указанной даты, сервер вернет код статуса «304 Not Modified».

Использование данного метода направлено на уменьшение нагрузки на сеть.

Согласование содержимого: это механизм автоматического определения необходимого ресурса при условии существования нескольких разнотипных версий документа. Субъектами могут быть не только документы, но и возвращаемые страницы с ошибками.

Различают 2 основных типа согласований:

- Управляемое сервером
- Управляемое клиентом.

Одновременно могут использоваться оба типа согласования или каждый из них в отдельности.

В спецификации выделяют третий тип согласования: прозрачное согласование, как комбинированный вариант двух типов согласования.

Данное согласование полностью прозрачно для сервера и клиента. В данном случае используется общий кэш, содержащий список вариантов, как для согласования управляемого клиентом, так и для согласования, управляемого сервером. В случае если кэш понимает все эти варианты, то он самостоятельно делает выбор. Это снижает нагрузки с сервера и помогает исключить дополнительные запросы со стороны клиента.

Управляемое сервером: при наличии нескольких версий ресурса сервер может анализировать заголовки запроса клиента, чтобы выдать ответ, наиболее подходящий для данного конкретного запроса.

Управляемое клиентом: в данном случае на стороне клиента определяется тип содержимого. Для этого сервер возвращает сообщение с кодом 300 (Multiple Choice) или 406 (Not Acceptable) список вариантов, среди которых пользователь ЭВМ выбирает наиболее походящий.

Конечно, данный протокол не изменился с 1999 года, но появилось множество новых настроек и расширений для этого протокола. О самых важных и востребованных расширениях речь пойдет дальше.

HTTPS – это расширение протокола HTTP, поддерживающее шифрование. Передаваемые по протоколу данные «упаковываются» в криптографический SSL протокол или TLS протокол. Этот протокол был впервые описан в 1994 году компанией Netscape Communications для браузера Netscape Navigator. Сейчас HTTPS широко распространен и используется во Всемирной паутине, также данная технология поддерживается большинством современных браузеров.

Принцип работы HTTPS предельно прост. Рассмотрим его более подробным образом. HTTPS не является отдельным протоколом. Это протокол HTTP, работающий через шифрование. Он защищает от атак, которые базируются на прослушивании сетевого соединения.

Обычно, HTTPS URL использует 443 TCP-порт (для незащищённого HTTP — 80). Чтобы подготовить сервер для обработки https-соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера. Сертификат состоит из 2 частей (2 ключей) — public и private. Публичная (public) часть ключа используется для шифрования трафика от клиента к серверу. Частная(private) часть ключа предназначена для дешифрования трафика, полученного от клиента на сервере.

После того как данные два ключа сгенерированы на основе публичной части ключа формируется запрос в центр сертификации, в ответ на которой центр высылает подписанный сертификат. Центр сертификации при подписании проверяет клиента, что позволяет однозначно удостоверять держателя ключа.

Также существует возможность создать такой ключ, не обращаюсь в центр сертификации. Данный прием характерен для серверов, работающий под управлением ОС семейства UNIX с помощью специальных программ. Подписывают данные сертификаты самим собой (самоподписанные сертификаты).

Также данная система сертификатов можно использовать для доступа только авторизованных пользователей на какой-либо сервер. Для этого, обычно, администратор сервера создает сертификаты для каждого пользователя в отдельности и загружает их в браузер каждого пользователя. Такой сертификат содержит имя и адрес электронной почты авторизованного пользователя. Каждый пользователь проверяется на наличие данного ключа при каждом соединении с сервером.

В HTTPS для шифрования трафика и данных используются ключи длиной 40,56,128 и 256 бит. Многие современные сайты требуют использование браузеров, которые поддерживают длину ключа от 128 бит, что обеспечивает достаточный уровень защищенности данных.

Дайджест аутентификация доступа является одним из широко распространенных методов, используемых веб-сервером для обработки учетных данных пользователей веб-браузера. Данный метод использует шифрование для отправки пароля через сеть, что повышает уровень защищенности и снижает риск «кражи» данного пароля.

Технически дайджест аутентификация представляет собой применение MD5 криптографического хэширования с использованием случайных значений для более трудной расшифровки кода злоумышленником. К сожалению, у дайджест аутентификации есть свои преимущества и недостатки. Рассмотрим их подробнее.

Схема дайджест аутентификации показана на Рис.2.

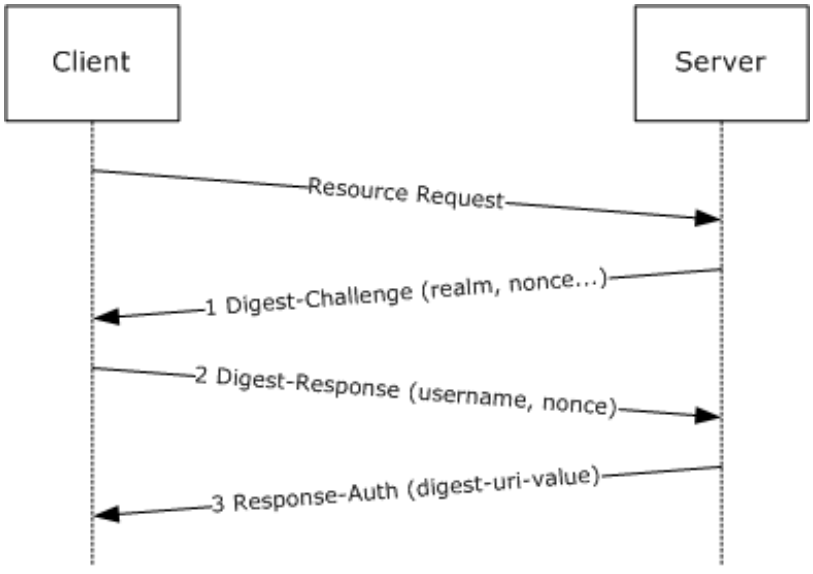

Рис. 2. Схема дайджест аутентификации

Преимущества:

- Пароль не используется непосредственно в дайджесте. Это позволяет в некоторых реализациях хранить пароль в закодированной форме, а не в виде открытого текста.
- Также было введено случайное некоторое значение, определяющееся на стороне клиента, которое позволяет предотвратить атаку уже заранее заготовленным способом.
- Серверное случайное значение может иметь временные метки. Поэтому сервер может проверить предоставленные клиентами случайные значения для предотвращения атак повторного воспроизведения.
- Сервер также может хранить таблицу недавно используемых случайных кодов, что предотвращает их повторное использование.

Недостатки:

- Если качество защиты не определено сервером, то клиент будет работать по умолчанию в пониженной защите.
- Дайджест аутентификация уязвима к атакам человека посередине. Например, злоумышленник может отправить клиентам инструкцию использовать базовую аутентификацию доступа. То есть дайджест аутентификация не предоставляет пользователем механизма проверки пользователя сервера.

В заключении хотелось бы сказать, в наше время трудно или практически невозможно обойтись без Всемирной паутины. Тысячи, миллионы, миллиарды людей взаимодействуют друг с другом по средствам Интернета. К сожалению, все они уязвимы к множеству атак, но при помощи новых методов шифрования их данные и трафик, могут находиться в безопасности. Конечно, понятие безопасность относительно, но с помощью таких технологий как HTTPS и дайджест аутентификация и многих других, их данные, с меньшим риском, могут быть украдены злоумышленниками.

Список литературы

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов. 3-е издание. СПб: Питер, 2008. 960 с.
2. Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети: учеб. пособие для вузов. М.: Изд-во МГТУ им.Н.Э.Баумана, 2003. 609 с.
3. Кенин А.М. Самоучитель системного администратора. 3-е издание. СПб: БХВ-Петербург, 2012. 512 с.